

EFFICIENT IMPLEMENTATION OF THE AI-REML ITERATION FOR VARIANCE COMPONENT QTL ANALYSIS

KATERYNA MISHCHENKO, SVERKER HOLMGREN, AND LARS RÖNNEGÅRD

ABSTRACT. Regions in the genome that affect complex traits, quantitative trait loci (QTL), can be identified using statistical analysis of genetic and phenotypic data. When restricted maximum-likelihood (REML) models are used, the mapping procedure is normally computationally demanding. We develop a new efficient computational scheme for QTL mapping using variance component analysis and the AI-REML algorithm. The algorithm uses an exact or approximative low-rank representation of the identity-by-descent matrix, which combined with the Woodbury formula for matrix inversion results in that the computations in the AI-REML iteration body can be performed more efficiently. For cases where an exact low-rank representation of the IBD matrix is available a-priori, the improved AI-REML algorithm normally runs almost twice as fast compared to the standard version. When an exact low-rank representation is not available, a truncated spectral decomposition is used to determine a low-rank approximation. We show that also in this case, the computational efficiency of the AI-REML scheme can often be significantly improved.

1. INTRODUCTION

Traits that vary continuously are called quantitative. In general, such traits are affected by an interplay between multiple genetic factors and the environment. Most medically and economically important traits in humans, animals and plants are quantitative, and understanding the genetics behind them is of great importance.

Date: February 11, 2008.

Key words and phrases. Quantitative Trait Mapping, restricted maximum-likelihood, average information matrix, identity-by-descent matrix, Woodbury formula, matrix inversion, truncated spectral decomposition, low-rank approximation, computational efficiency, cpu time.

The dissection of quantitative traits is generally performed by statistical analysis of genetic and phenotypic data for experimental populations. Such analysis can reveal quantitative trait loci, QTL, in the genome that affect the trait. In the field of QTL analysis (Lynch and Walsh, 1998), the use of variance component models are commonly used, see e.g. (Blangero et al., 2001). Using such models, the statistical analysis is performed using maximum-likelihood (ML) or restricted maximum-likelihood (REML) estimators which have the advantage that they do not use specific assumptions on the design or balance of the data. These types of schemes are often considered to be statistically efficient since they utilize all available data. On the other hand, the corresponding algorithms are also relatively computationally demanding since non-linear optimization problems must be solved using an iterative procedure. When searching for the most likely locations of the QTL, REML optimization problems must be solved for many genetic locations within an outer global optimization procedure. Furthermore, if the significance of the results is established using an experimental procedure like parametric bootstrap (Davison and Hinkley, 1997), hundreds or thousands of QTL searches must be performed. This implies that the computational complexity for the numerical methods for solving the REML problems must be minimal for the QTL mapping to be viable.

During the last two decades, specialized algorithms for variance component analysis in animal breeding settings have been developed and implemented in codes like ASREML, DMU and VCE, see (Druet and Ducrocq, 2006) and references therein. The algorithms used in these codes have in common that they use a specific Newton-type iteration, the *average information restricted maximum likelihood algorithm*, AI-REML. However, the structure of the problems in traditional animal breeding applications is different from that of QTL analysis problems, and the computational schemes should be examined and possibly modified before they are applied in a QTL analysis setting. In this short paper we perform this type of investigation and develop a new efficient computational scheme for QTL mapping using variance component analysis and the AI-REML algorithm. This type scheme should then be applied within an efficient and robust optimization method for maximizing the likelihood in the AI-REML method, and also within an efficient and

robust global optimization algorithm for the search for the outer optimization problem where the most likely position of the QTL is determined.

2. THE AI-REML ALGORITHM

A general linear mixed model is given by

$$(1) \quad y = Xb + Rr + e,$$

where y is a vector of n observations, X is the $n \times n_f$ design matrix for n_f fixed effects, R is the $n \times n_r$ design matrix for n_r random effects, b is the vector of n_f unknown fixed effects, r is the vector of n_r unknown random effects, and e is a vector of n residuals. In the QTL analysis setting, we assume that the entries of e are identically and independently distributed and there is a single observation for each individual in the pedigree. In this case, the covariance matrices are given by $\text{var}(e) = I\sigma_e^2$ and $\text{var}(r) = A\sigma_a^2$, where A is referred to as the *identity-by-descent* (IBD) matrix. Using these assumptions, we have that

$$(2) \quad \text{var}(y) \equiv V = \sigma_a^2 A + \sigma_e^2 I \equiv \sigma_1 A + \sigma_2 I.$$

Here, $\sigma_1 \geq 0$ and $\sigma_2 > 0$ and we assume that the phenotype follows a normal distribution with $y \sim MVN(Xb, V)$. At least two different procedures can be used for computing estimates of b and $\sigma_{1,2}$. In the standard codes mentioned above, the parameters are computed from the mixed-model equations (MME) (Lynch and Walsh, 1998) using the inverse of A . To be able to use this approach, A has to be positive definite, or it must be modified so that this property holds. Also, the computations can be performed efficiently if A is sparse. For the QTL analysis problems, the IBD matrix A is often only semi-definite and not necessarily sparse. Here the values of $\sigma_{1,2}$ are given by the solution of the minimization problem

$$(3) \quad \text{Min } L,$$

$$\text{s.t. } \sigma_1 \geq 0$$

$$(4) \quad \sigma_2 > 0$$

where L is the log-likelihood for the model (1),

$$(5) \quad L = -2\ln(l) = C + \ln(\det(V)) + \ln(\det(X^T V^{-1} X)) + y P^T y,$$

and the projection matrix P is defined by

$$(6) \quad P = V^{-1} - V^{-1} X (X^T V^{-1} X)^{-1} X^T V^{-1}.$$

In the original AI-REML algorithm, the minimization problem is solved using the standard Newton scheme but where the Hessian is substituted by the *average information matrix* H^{AI} , whose entries are given by

$$(7) \quad H_{i,j}^{AI} = y^T P \frac{\partial V}{\partial \sigma_i} P \frac{\partial V}{\partial \sigma_j} P y \quad , \quad i, j = 1, 2.$$

The entries of the gradient of L are given by

$$(8) \quad \frac{\partial L}{\partial \sigma_i} = \text{tr} \left(\frac{\partial V}{\partial \sigma_i} P \right) - y^T P \frac{\partial V}{\partial \sigma_i} P y \quad , \quad i = 1, 2.$$

3. EFFICIENT IMPLEMENTATION OF AI-REML OPTIMIZATION

The main result of this paper is an algorithm that allows for an efficient implementation of the iteration body in the AI-REML algorithm for variance component analysis in QTL mapping problems. In (Mishchenko et al.), we examine optimization schemes for the REML optimization and different approaches for introducing the constraints for $\sigma_{1,2}$ in the AI-REML scheme. For cases when the solution to the optimization problem is close to a constraint, different ad-hoc fixes have earlier been attempted to ensure convergence for the Newton scheme. In (Mishchenko et al.), we show that by introducing an active-set formulation in the AI-REML optimization scheme, robustness and good convergence properties were achieved also for cases when the solution is close to a constraint. When computing the results below, we use the active-set AI-REML optimization scheme from (Mishchenko et al.).

From the formulas in the previous section, it is clear that the most computationally demanding part of the AI-REML algorithm is the computation of the matrix P , and especially the explicit inversion of the matrix $V = \sigma_1 A + \sigma_2 I$, where A is a constant

semi-definite matrix and $\sigma_{1,2}$ are updated in each iteration. If V is regarded as a general matrix, i.e. no information about the structure of the problem is used, a standard algorithm based on Cholesky factorization with complete pivoting is the only alternative, and $O(n^3)$ arithmetic operations are required. If A is very sparse, the work can be significantly reduced by using a sparse Cholesky factorization, possibly combined with a reordering of the equations. This can be compared to the approach taken e.g. in (Bates, (2004)), where a sparse factorization is used for A when solving the mixed-model equations. However, as remarked earlier A is only semi-definite and not very sparse for the QTL analysis problems and this approach is not an option here.

The IBD matrix A is a function of the location in the genome where the REML model is applied. The key observation leading to a more efficient algorithm for inverting V is that the rank of A at a location with complete genetic information only depends on the size of the base generation. At such locations, A will be a rank- k matrix where $k \ll n$ in experimental pedigrees with small base generations. If the genetic information is not fully complete, A can still be approximated by a low-rank matrix. The error in such an approximation can be made small when the genetic distance to a location with complete data is small.

Let a symmetric rank- k representation of A be given by

$$(9) \quad A = ZZ^T,$$

where Z is an $n \times k$ matrix. We now exploit this type of representation to compute V^{-1} .

Case 1: An exact low-rank representation of A is available a-priori. In general, the inverse of a matrix plus a low-rank update can be computed using the Woodbury formula, see e.g. (Golub and C.Van Loan, (1996)),

$$(10) \quad B^{-1} = (C + S_1 S_2^T)^{-1} = C^{-1} - C^{-1} S_1 (I + S_2^T C^{-1} S_1)^{-1} S_2^T \cdot C^{-1}$$

Applying (10) with $C = I\sigma_2$, $S_1 = Z$ and $S_2 = \sigma_1 Z$, we get the following formula for computing V^{-1} ,

$$(11) \quad V^{-1} = I\sigma_2^{-1} - \sigma_2^{-1}\hat{\sigma}Z(I + \hat{\sigma}Z^T Z)^{-1}Z^T,$$

where $\hat{\sigma} = \sigma_1\sigma_2^{-1}$. The $k \times k$ matrix $Z^T Z$ can be computed once, before the Newton iteration is started. Also, the matrix $(I + \hat{\sigma}Z^T Z)$ is only of size $k \times k$, and its inverse can be computed using a standard factorization method in $O(k^3)$ arithmetic operations.

Case 2: No low-rank representation of A is given a-priori. If only the matrix A is given, a low-rank representation/approximation can be computed once before the Newton iteration is started. Then, the Woodbury formula is again used to get an efficient computational algorithm for the iterations. Computing a low-rank approximation of a matrix is a problem with a long history (Eckart and Young, 1936), and the standard tool is the truncated singular value decomposition, see e.g. (Golub and C.Van Loan, (1996); Press et al., 1997), which computes the optimal approximation in Frobenius norm. In our case, A is symmetric and positive semi-definite, and the SVD is equivalent to the standard spectral decomposition. Also, the spectral decomposition can be written as $A = W\Lambda W^T = (W\sqrt{\Lambda})(W\sqrt{\Lambda})^T$, where $\Lambda = \text{diag}(\lambda_i)$, $i = 1, \dots, n$. Hence, we compute a low-rank representation of A using a truncated spectral decomposition,

$$(12) \quad A = (W_t\sqrt{\Lambda_t})(W_t\sqrt{\Lambda_t})^T,$$

where Λ_t is a diagonal matrix of k eigenvalues and W_t is a rectangular matrix of size $n \times k$ containing the corresponding orthogonal eigenvectors. Here, Λ_t and W_t are obtained by eliminating the $n - k$ smallest eigenvalues and the corresponding eigenvectors from the standard spectral decomposition. Such idea is used in Genetic Principal Component Analysis, see e.g. (Kirpatrick and Meyer, 2004).

Again using the Woodbury formula (10), now with $C = I\sigma_2$, $S_1 = W_t\sqrt{\Lambda_t}$ and $S_2 = \sigma_1 W_t\sqrt{\Lambda_t}$, we compute \tilde{V}^{-1} according to,

$$(13) \quad \tilde{V}^{-1} = I\sigma_2^{-1} - \sigma_2^{-1}\hat{\sigma}(W_t\sqrt{\Lambda_t})(I + \hat{\sigma}\Lambda_t)^{-1}(W_t\sqrt{\Lambda_t})^T.$$

Or in equivalent form,

$$(14) \quad \tilde{V}^{-1} = I\sigma_2^{-1} - \sigma_2^{-1}\hat{\sigma}W_t(\Lambda_t^{-1} + \hat{\sigma}I)^{-1}W_t^T.$$

A similar approach for matrix inversion has been used for problems in Gaussian process regression with applications in e.g. machine learning, see (Schwaighofer and Tresp, 2002; Rasmussen and Williams, 2005). However, as far as we know, an approach based on a truncated spectral decomposition and formula (13) has not been pursued for Newton iterations for REML-problems before.

When performing the computation described by (14), the matrix W_t is constant and can be precomputed before the iteration starts. The matrix $(\Lambda_t^{-1} + \hat{\sigma}I)$ is diagonal, so its inverse is trivially computable.

The factorization (12) is a rank- k representation of A . This is exact if A has at least $n - k$ zero eigenvalues, and then $\tilde{V}^{-1} = V^{-1}$. For a general IBD matrix A , more than k eigenvalues will normally be non-zero, resulting in that \tilde{V}^{-1} is an approximation of V^{-1} . The accuracy of this approximation depends on the quality of the truncated factorization (12) for A . For the QTL analysis problems, we need to choose the rank k such that the accuracy of the solution to the REML problem, i.e. the variance components $\sigma_{1,2}$, is sufficient.

Assume that the eigenvalues are ordered such that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Several criteria for choosing k can be considered. One option is to simply set k to a predetermined value, either motivated by the known value of the rank at a fully informative position in the genome or determined as a given percentage of n . Such a choice has the advantage that it is possible to pre-determine the computational work needed for computation of \tilde{V}^{-1} in (14). Another easily applied criterion for determining k is given by

$$(15) \quad k = k(\tau) = \{\max i : \lambda_i \geq \tau\lambda_1\},$$

where the threshold τ is for example chosen as $\tau = 0.001$. This means that all eigenvalues of the matrix A that are smaller than $0.001\lambda_1$ are neglected and the corresponding

columns of W in the spectral decomposition are deleted. In the numerical experiments presented in the next section, we use the truncation criterion (15).

For computing the truncated spectral decomposition, we currently use a standard eigenvalue solver, compute all eigenvalues and eigenvectors, and then ignore the data connected to eigenvalues smaller than given by the truncation criterion. For problems with large data matrices where it is known that only a small number of eigenvalues will be large, an alternative can be to use e.g. the Lanczos iteration for computing these. Potentially, this can reduce the arithmetic work for computing the truncated decomposition before the iteration starts.

Once V^{-1} has been computed using either (14) or (11), formula (6) is used to compute P , and the entries of the Hessian H^{AI} and the gradient of L are computed using the formulas (7) and (8). The matrices involved have common terms which are computed once at the beginning of the current iteration to reduce the total complexity.

4. NUMERICAL RESULTS

In this section we present numerical experiment computed using experimental IBD matrices of size 767×767 drawn from a chicken population (Kerje et al., 2003). For these first pilot investigations of the computational performance we have chosen to implement the algorithms in Matlab, and we present execution time measurements for the Matlab scripts. At a later stage, we will implement the most promising algorithms also in C, which will reduce the execution time further. In this case, we expect that the relative gain from using the new algorithms will be larger than indicated below. Matlab's native implementation of standard inversion is presumably more efficient than our Matlab scripts for the new algorithms, which are interpreted at runtime.

Case 1: We first present numerical results for the case when the IBD matrices A have exact low-rank representations given a-priori. Let T_C be the measured computational time for the computation of V^{-1} within the AI-REML scheme using standard inversion, and let T_W be the corresponding time when using the low-rank representation and

formula (11). In Figure 1 we show T_C/T_W , i.e. the speedup for the computations of V^{-1} , as a function of the rank k of the matrices A .

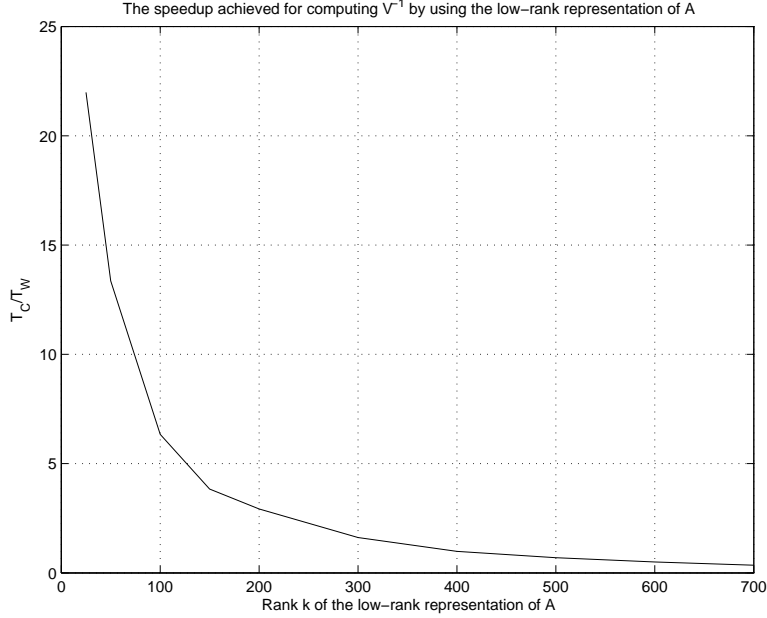


FIGURE 1. The speedup achieved for the computations of V^{-1} in the AI-REML scheme achieved by using the scheme based on (11) instead of standard inversion

From Figure 1, it is clear that for IBD matrices of size 767×767 , the implementation using the low-rank representation is faster if the rank k is smaller than approximately 350. Also, for $k < 50$, the speedup is more than one order of magnitude.

In Table 1 we present the total CPU-times for the AI-REML scheme and the CPU-time for the computations of V^{-1} within this scheme for different values of k and number of iterations equal to 9. The total CPU-times for the AI-REML are denoted by T_C^{tot} and T_W^{tot} , while the CPU-times for the computation of V^{-1} are denoted by T_C and T_W .

From Table 1 it is clear that for the problems studied here, computing V^{-1} using standard inversion accounts for approximately half of the computational time in the AI-REML scheme. For small values of k , the time needed for inverting V^{-1} using the low-rank representation can effectively be ignored, and the improved algorithm runs approximately twice as fast as the original scheme using standard inversion.

k	T_C^{tot}	T_C	T_W^{tot}	T_W
700	10.28	5.45	20.3	15.5
600	10.28	5.45	15.7	10.9
500	10.28	5.45	12.6	7.81
400	10.28	5.45	10.4	5.55
300	10.28	5.45	8.20	3.37
200	10.28	5.45	6.62	1.86
150	10.28	5.45	6.25	1.42
100	10.28	5.45	5.69	0.86
50	10.28	5.45	5.23	0.41
25	10.28	5.45	5.08	0.25

TABLE 1. Total CPU-time for the AI-REML computations and CPU-time for computing V^{-1} for the schemes using standard inversion and a low-rank representation

Case 2: We now turn to case 2. Here a low-rank representation of A is not available a-priori, and a truncated eigenvalue decomposition must be computed before the AI-REML iteration starts. We present results from computations performed using 18 different IBD matrices corresponding to genetic locations where a low-rank representation is not given. The first set of results shows the computational time $T_{C,TW}$ for computing V^{-1} within the AI-REML scheme. In this case we compare standard inversion (C) to the scheme given by (13), based on truncated spectral decomposition (TW).

In Table 2, we show the CPU-time and the number of AI-REML iterations for the two schemes. For the scheme based on the truncated spectral decomposition, the table also shows the rank k as determined by the truncation criterion (15) and the relative errors (in %) for the variance components ($e_{\sigma_{1,2}}$) arising from using a low-rank approximation of A . For the matrices A_{11} and A_{12} , the variance component σ_1 is zero, and it is not possible to compute the relative error. For these matrices, the absolute errors in

A	C		TW									
	T_C	# it	$\tau = 0.001$					$\tau = 0.005$				
			T_{TW}	k	#it	e_{σ_1} [%]	e_{σ_2} [%]	T_{TW}	k	# it	e_{σ_1} [%]	e_{σ_2} [%]
A_1	4.18	7	2.20	294	7	0.82	1.67	0.531	82	7	6.7	4.7
A_2	4.16	7	1.12	166	7	0.03	0.377	0.203	37	6	7.3	1.8
A_3	3.50	6	1.25	169	7	0.11	1.19	0.307	35	8	12.7	3.0
A_4	3.48	6	1.87	280	6	1.14	1.83	1.05	71	15	40.8	7.5
A_5	2.81	5	1.81	291	6	0.94	1.56	0.920	82	11	30.4	5.4
A_6	4.89	8	2.60	294	8	0.64	0.76	0.438	81	6	8.1	3.1
A_7	2.88	5	1.66	286	5	0.84	0.53	0.313	70	5	5.9	2.1
A_8	4.93	8	1.22	163	8	0.44	0.18	0.314	43	7	3.9	0.64
A_9	8.36	13	0.671	54	12	0.04	0.03	0.328	15	13	4.5	0.09
A_{10}	15.3	23	1.22	50	23	0.07	0.02	0.527	10	23	0.05	0.05
A_{11}	1.98	4	0.140	46	4	-	0	0.093	13	4	-	0
A_{12}	2.00	4	0.344	71	5	-	0	0.108	14	5	-	0
A_{13}	5.45	9	0.467	57	9	0.004	0	0.173	12	9	0.02	0.02
A_{14}	2.86	5	0.873	192	5	0.53	0.26	0.327	51	7	16.9	1.6
A_{15}	2.77	5	0.884	198	5	0.36	0.41	0.250	51	5	3.1	1.1
A_{16}	2.83	5	0.749	173	5	0.19	0.39	0.327	44	8	22.2	2.0
A_{17}	2.80	5	1.28	270	5	0.96	0.95	0.311	60	6	6.3	3.2
A_{18}	4.23	7	2.62	303	8	0.69	0.91	0.485	86	6	8.0	3.7

TABLE 2. CPU-time for computing V^{-1} within the AI-REML iteration,
and errors arising from exploiting a low-rank approximation of A

σ_1 is very small for both values of τ . From the results in Table 2, it is clear that the accuracy of the variance components does not only depend on the value of the parameter τ , but also on the properties of the matrix A . When an approximation of V^{-1} is employed in the computations, the optimization landscape is affected in different ways for different IBD matrices A , resulting in different behavior for the optimization

scheme and different errors in the location of the optima. However, for the problems studied here, choosing $\tau = 0.001$ results in that the variance components are determined with sufficient accuracy for all the IBD matrices examined.

In the next set of experiments we study the speedup and error introduced by using an approximate inverse of V in some more detail for the two IBD matrices A_1 and A_{13} . As can be seen from Table 2, A_1 is harder to approximate with a low rank matrix than A_{13} . In Tables 3 and 4, we vary the value of the parameter τ in the truncation criterion (15) and show the resulting rank k of the approximation of A , the number of iterations in the AI-REML optimization scheme, the speedup T_C/T_{TW} , and the relative errors in the variance components.

τ	k	# it	T_C/T_{TW}	e_{σ_1} [%]	e_{σ_2} [%]
10^{-7}	699	7	0.46	0.000	0.000
10^{-6}	662	7	0.51	0.002	0.001
10^{-5}	556	7	0.71	0.070	0.054
10^{-4}	294	7	1.97	0.82	1.67
$5 \cdot 10^{-4}$	82	7	7.68	6.75	4.69
10^{-3}	29	36	3.28	95.2	6.70
$5 \cdot 10^{-3}$	-	no conv.	-	-	-

TABLE 3. Results for the IBD matrix A_1 for different values of the truncation parameter τ

In Figure 2 we finally present total timings for solving the AI-REML problems using the new scheme exploiting a truncated spectral decomposition, including the time needed for computing the truncated factorization prior to the AI-REML iterations. We also compare these timings to the corresponding results for the standard scheme using direct inversion. In the figure, the timings for all matrices $A_1 - A_{18}$ are shown as a function of the number of AI-REML iterations required. When several matrices result in the same number of iterations, the cpu time for each matrix and the average result are shown.

τ	k	# it	T_C/T_{TW}	e_{σ_1} [%]	e_{σ_2} [%]
10^{-7}	134	9	6.41	0.000	0.000
10^{-6}	134	9	6.50	0.000	0.000
10^{-5}	123	9	7.12	0.000	0.000
10^{-4}	57	9	15.0	0.004	0.006
$5 \cdot 10^{-4}$	12	9	34.6	0.009	0.025
10^{-3}	9	9	37.3	0.024	0.025
$5 \cdot 10^{-3}$	7	9	44.7	0.109	0.027
10^{-2}	5	9	55.7	2.3	0.046
$5 \cdot 10^{-2}$	-	no conv.	-	-	-

TABLE 4. Results for the IBD matrix A_{13} for different values of the truncation parameter τ

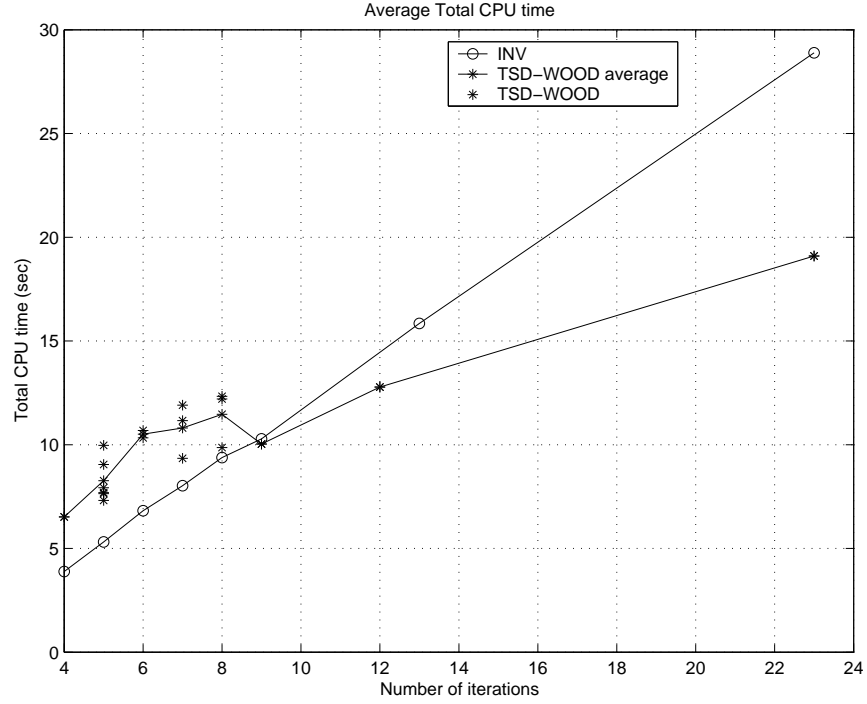


FIGURE 2. Total CPU-times for solving the AI-REML problems using the scheme based on standard inversion of V and the new scheme using a truncated spectral decomposition

Theoretically, the average cpu time for the scheme exploiting the truncated spectral decomposition should be a straight line. Deviations are due to the inconsistency of time measurement in Matlab.

From Figure 2, we see that the standard scheme using direct inversion is faster when number of iterations is small, i.e. less than 9 iterations. The reason for this is that the spectral decomposition of the matrix A is relatively costly, and it must be amortized over a number of iterations before the faster iterations begin to pay off. When the number of iterations required is significantly larger than 9, which is often the case for real-world problems, the new algorithm is significantly faster also when no low-rank representations of the IBD matrices are not available a priori.

5. CONCLUSIONS

In this paper we present a family of algorithms that allow for an efficient implementation of the iteration body in the AI-REML algorithm for variance component analysis in QTL mapping problems. Combined with the improved optimization scheme in (Mishchenko et al.), the new algorithms form a basis for an efficient and robust AI-REML scheme for evaluating variance component QTL models.

The most costly operation in the AI-REML iteration body is the explicit inversion of the matrix $V = \sigma_1 A + \sigma_2 I$, where the IBD matrix A is constant and positive semi-definite, and $\sigma_1 \geq 0$ and $\sigma_2 > 0$ are updated in each iteration. The key information enabling the introduction of improved algorithms is that the rank of A at a location in the genome with complete genetic information only depends on the size of the base generation. At such locations, A will be a rank- k matrix where $k \ll n$, and by exploiting the Woodbury formula the inverse of V can be computed more efficiently than by using a standard algorithm based on Cholesky factorization. If the genetic information is not fully complete, a general IBD matrix A can still be approximated by a low-rank matrix and the error in such an approximation can be made small when the genetic distance to a location with complete data is small. More importantly, there might be a possibility for setting up a low-rank representation of the matrix A also at genetic

locations where the information is not complete. This is a topic of current investigation (Rönnegård and Carlborg, 2007).

We present results for IBD matrices A from a real data set for two different settings; Firstly, we show that if a low-rank representation of A is available a-priori, the inversion of V using the new algorithms is performed faster than using a standard algorithm if the rank k is smaller than approximately 350. Also, for $k < 50$, the speedup is more than one order of magnitude.

Then we also show that, even if a low-rank representation is not directly available and a low-rank approximation of A needs to be computed before the AI-REML iterations, significant speedup of the variance component model computations can still be achieved. For QTL mapping problems, the efficiency of our new method will increase when the ratio between the total pedigree size and base generation size increases, the density and informativeness of markers increases. Hence, the relative efficacy of the method will continuously increase in the future with deeper pedigrees and more markers. Also, we are currently developing a scheme for directly constructing a low-rank representation of the IBD matrix A also between markers, which will result in that the eigenvalue factorization prior to the AI-REML iterations is not needed any more.

REFERENCES

- Bates, D. (2004). *Sparse matrix representations of linear mixed models*. Technical Report. <http://www.stat.wisc.edu/bates/reports/MixedEects.pdf>.
- Blangero, J., J.T. Williams, and L. Almasy. 2001. *Variance component methods for detecting complex trait loci*, *Advances in Genetics* **42**, 151–181.
- Davison, A. C. and D. V. Hinkley. 1997. *Bootstrap methods and their application.*, Cambridge University Press, U.K.
- Druet, T. and V. Ducrocq. 2006. *Innovations in software packages in quantitative genetics*. World Congress on Genetics Applied to Livestock Production, Belo Horizonte, Brazil.
- Eckart, C. and G. Young. 1936. *The approximation of one matrix by another of lower rank*, *Psychometrika* **1**, no. 3, 211–218.
- Golub, G. and C. Van Loan. (1996). *Matrix Computations*, Third edition, The Johns Hopkins University Press.

- Kerje, S., Ö. Carlborg, L. Jacobsson, K. Schutz, C. Hartmann, P. Jensen, and L. Andersson. 2003. *The twofold difference in adult size between the red junglefowl and white leghorn chickens is largely explained by a limited number of qtls*, *Animal Genetics* **34**, 264–274.
- Kirpatrick, M. and K. Meyer. 2004. *Direct estimation of genetic principal components: Simplified analysis of complex phenotypes*, *Genetics* **168**, 2295–3206.
- Lynch, M. and B. Walsh. 1998. *Genetics and analysis of quantitative traits*, Sinauer Associates, Inc.
- Mishchenko, K., S. Holmgren, and L. Rönnegård. *Newton-type methods for reml estimation in genetic analysis of quantitative traits*. <http://arxiv.org/abs/0711.2619>.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1997. *Numerical recipes in c. the art of scientific computing*, Second edition, Cambridge University Press.
- Rasmussen, C.E. and C. Williams. 2005. *Gaussian processes for machine learning*, MIT Press.
- Rönnegård, L. and Ö. Carlborg. 2007. *Separation of base allele and sampling term effects gives new insights in variance component qtl analysis*, *BMC Genetics* **8**, no. 1.
- Schwaighofer, A. and V. Tresp. 2002. *Transductive and inductive methods for approximate gaussian process regression*, NIPS.

DEPARTMENT OF MATHEMATICS AND PHYSICS, MÄLARDALEN UNIVERSITY, BOX 883, SE-721 23
VÄSTERÅS, SWEDEN

E-mail address: `kateryna.mishchenko@mdh.se`

DIVISION OF SCIENTIFIC COMPUTING, DEPARTMENT OF INFORMATION TECHNOLOGY, UPPSALA
UNIVERSITY, SWEDEN

E-mail address: `sverker@it.uu.se`

LINNÆUS CENTER FOR BIOINFORMATICS, UPPSALA UNIVERSITY, SWEDEN

E-mail address: `lars.ronnegard@lcb.uu.se`